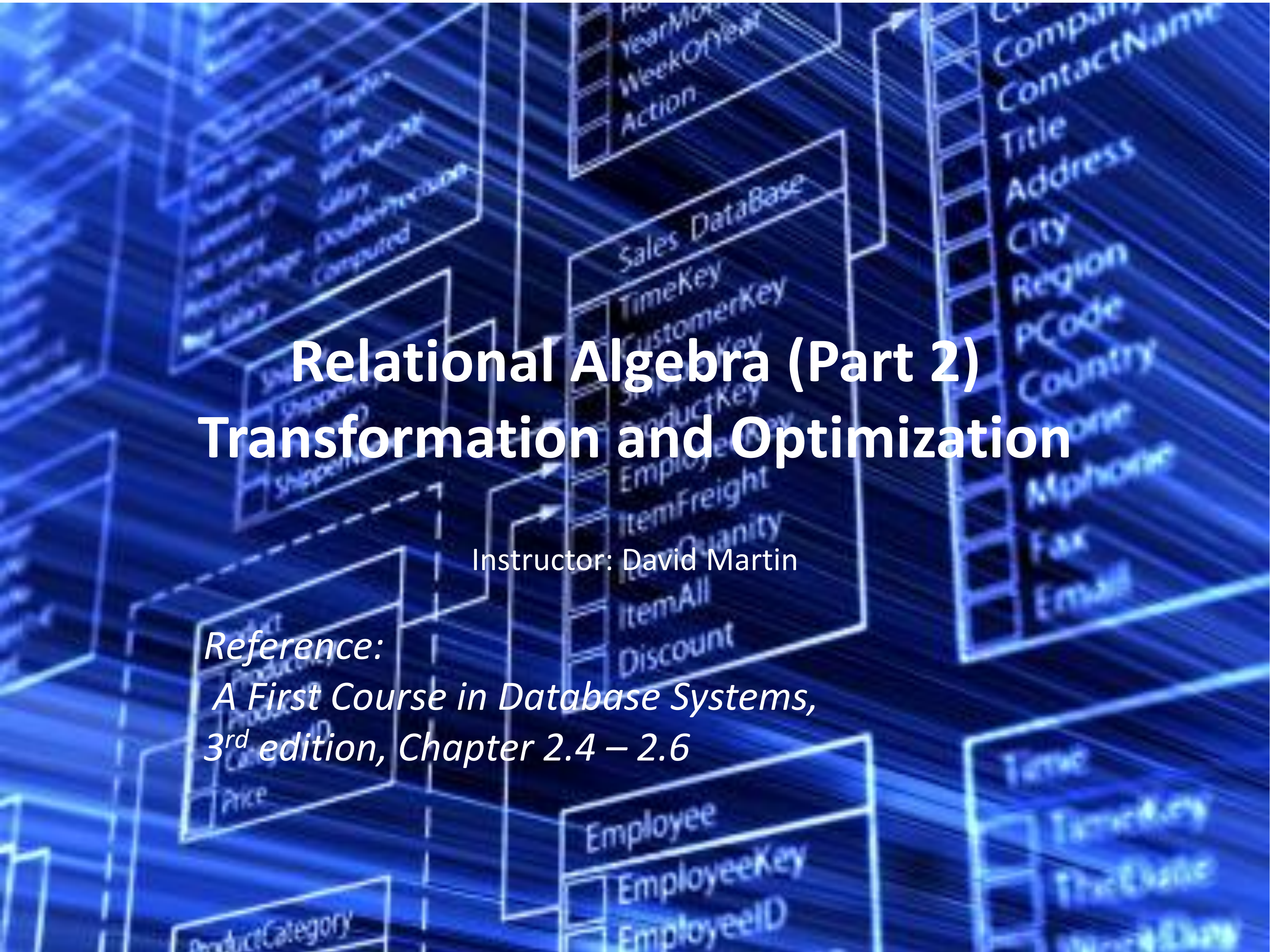


Relational Algebra (Part 2) Transformation and Optimization

Instructor: David Martin

Reference:
A First Course in Database Systems,
3rd edition, Chapter 2.4 – 2.6

Independence of Basic Operators

- Many interesting queries can be expressed using the five basic operators (σ , π , x , U , - , ρ)
- Can one of the five operators be derived by the other four operators?

<u>Theorem (Codd):</u>

The five basic operators are independent of each other. In other words, for each relational operator *o*, there is no relational algebra expression that is built from the rest that defines *o*.

- x
- π
- σ
- U
- -
- ρ

More Complex Queries

- Relational operators can be composed to form more complex queries. We have already seen examples of this in SQL.

Enrollments(esid, ecid, grade)

Courses(cid, cname, instructor-name)

- Query 1: Find the student id, grade and instructor where the student had a grade that was more than 80 points in a course.

$$\sigma_{\text{grade>80}} \; (\; \pi_{\text{esid, grade, instructor-name}} \; ($$

$$\sigma_{\text{Enrollments.ecid = Courses.cid}} \; (\text{Enrollments x Courses}) \;) \;)$$

Query 2

Enrollments(esid, ecid, grade)

Courses(cid, cname, instructor-name)

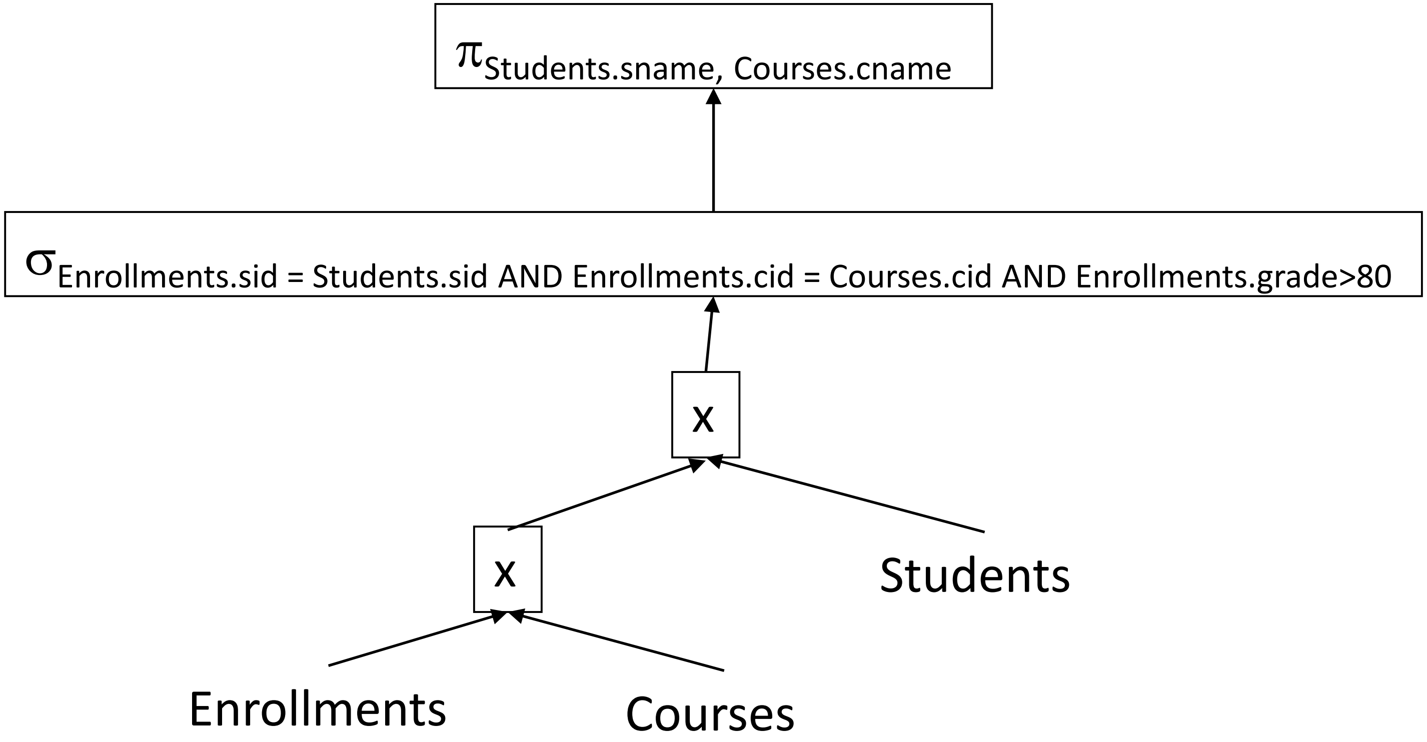
Students(sid, sname)

- Find the student name and course name where the student had a grade that was more than 80 points in a course.

$$\pi_{\text{Students.sname, Courses.cname}} \Big(\sigma_{\substack{\text{Enrollments.ecid = Courses.cid} \\ \text{AND Enrollments.esid = Students.sid} \\ \text{AND Enrollments.grade > 80}}} (\text{Enrollments x Courses x Students}) \Big)$$

An Execution Plan for Query 2

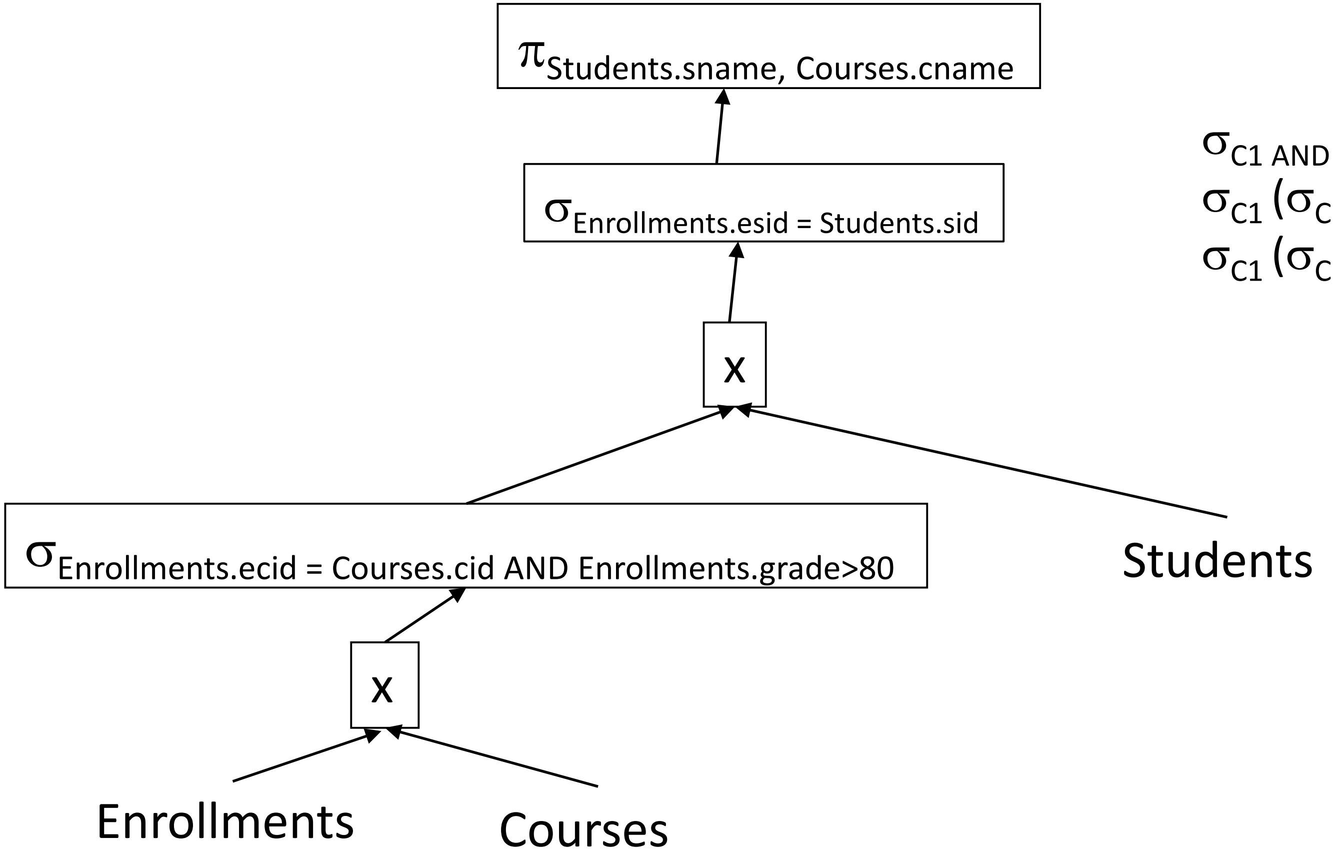
- Find the student name and course name where the student had a grade more than 80 points in a course.



$\pi_{\text{Students.sname, Courses.cname}}$
$\sigma_{\text{Enrollments.sid = Students.sid AND Enrollments.cid = Courses.cid AND Enrollments.grade>80}}$
x
x
Students
Enrollments
Courses

Another Execution Plan for Query 2

- Find the student name and course name where the student had a grade more than 80 points in a course.



$\pi_{\text{Students.sname, Courses.cname}}$

$\sigma_{\text{Enrollments.esid = Students.sid}}$

x

$\sigma_{\text{Enrollments.ecid = Courses.cid AND Enrollments.grade>80}}$

Students

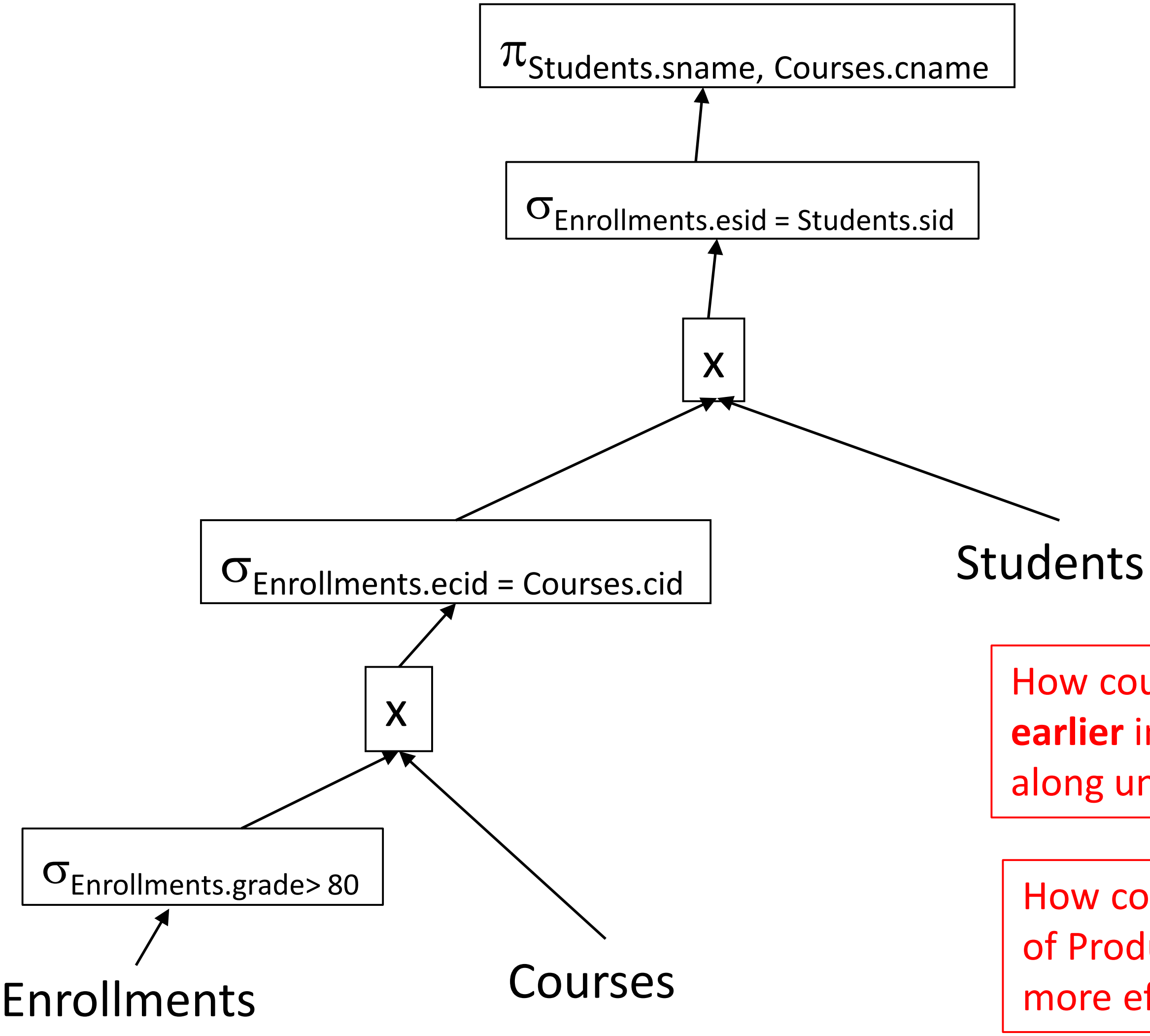
x

Enrollments

Courses

$\sigma_{\text{C1 AND C2 AND C3}}$ (E x C x S) =
σ_{C1} ($\sigma_{\text{C2 AND C3}}$ (E x C x S)) =
σ_{C1} ($\sigma_{\text{C2 AND C3}}$ (E x C) x S)

A Third Execution Plan for Query 2



$\pi_{\text{Students.sname, Courses.cname}}$

$\sigma_{\text{Enrollments.esid = Students.sid}}$

x

$\sigma_{\text{Enrollments.ecid = Courses.cid}}$

Students

x

$\sigma_{\text{Enrollments.grade> 80}}$

Enrollments

Courses

How could we do projections **earlier** in plan to avoid carrying along unnecessary attributes?

How could we do **Joins**, instead of Products to make plan more efficient?

Execution Plans

(Out-of-Scope for Exams)

- When do you do SELECTION?
  - Predicate pushdown is always a good idea.
- How do you access each table?
  - Scan, index (which index), hash, …
- What’s the order in which you Join tables?
  - Join/Equi-join is common; avoid Cartesian product
  - But which table do you start with?
    - Predicates on indexed columns are often useful in picking first table, then next table, to join, …
- What join method do you use for each join?
  - Nested loop join, merge join, hash-join, …
- How much parallelism do you use?
  - How do you schedule tasks to hardware?
- Do you need to sort? If so, when do you sort?

Query Optimization

- Comparing Execution Plans and finding a “good” (not necessarily best) plan
- Statistics that DBMS may keep to help calculate approx. query cost
  - Cardinality (number of rows) in table
  - Highest and lowest (non-null) value in column
  - Column cardinality (number of different values in column)
  - Number of appearances of the top 10 most frequent value in each column
  - Join cardinality between tables for particular equ-join
    - May be calculated, not stored; not well-defined if there are conditions (predicates) on the tables
  - Many other statistics are calculated approximately
- How frequently are stored statistics updated?
- Cost: CPU? I/O? Network? How do these get combined to compare?

EXPLAIN Statement

- Shows information about query plan
  - Each DBMS that has EXPLAIN has its own variation
  - Try it with PostgreSQL
- You may want to try to rewrite query yourself to find better execution plan if Query Optimizer isn’t smart enough to do so
- Should Optimizer take advice from users?